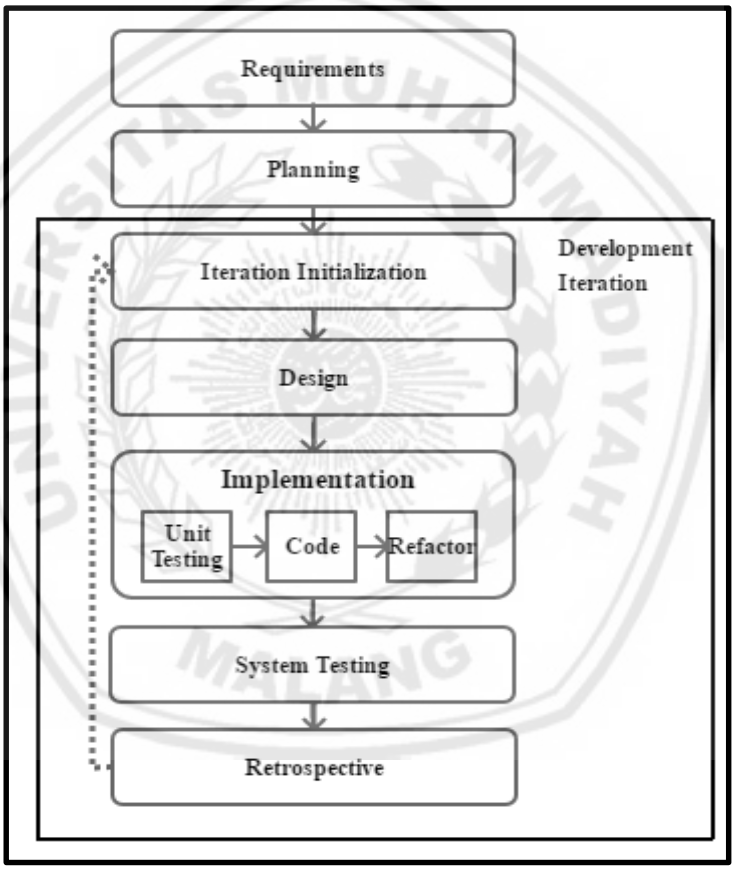
BAB III
METODOLOGI PENELITIAN

Pada bab ini dijelaskan garis besar langkah-langkah penelitian yang akan dilakukan. Metode penelitian ini disusun agar penelitian fokus pada rencana awal. Metodologi pengembangan aplikasi menggunakan *Personal Extreme Programming* (PXP). Proses PXP memiliki tahap yang harus dikerjakan dan dipenuhi sesuai [7]. Tahap-tahap tersebut ditunjukkan pada gambar 3.1.



Gambar 3.1. Tahap-tahap Metodologi PXP [7].

Gambar 3.1 menunjukkan tahap-tahap pengembangan aplikasi menggunakan metodologi PXP yang dilalui mulai dari *requirements* hingga *retrospective*. Aplikasi perpustakaan Kejaksaan Negeri pada penelitian ini akan dibangun menggunakan bahasa pemrograman PHP serta menggunakan *database* MySQL dan *framework* CodeIgniter. Ada beberapa tahapan dalam membangun aplikasi perpustakaan

Kejari Batu menggunakan PXP yaitu *requirements, planning, iteration initialization, design, implementation, system testing,* dan *retrospective*.

Requirements sebagai tahap pengumpulan data kebutuhan awal klien. Kebutuhan-kebutuhan tersebut dilakukan perencanaan lebih lanjut pada tahap *planning*. Tujuannya untuk menentukan estimasi waktu pengerjaan tiap kebutuhan yang diperoleh pada tahap *requirements*, kebutuhan mana yang harus diprioritaskan, dan membagi kebutuhan dalam iterasi-iterasi. Tahap *iteration initialization* adalah sebagai awal tiap perulangan implementasi membangun sistem. Tahap *design* PXP berbeda dengan metodologi pengembangan *software* yang lain, dimana pengembang hanya membuat desain sederhana berdasarkan kebutuhan klien yang diperoleh pada tahap *requirements* saja tanpa membuat desain yang mungkin dibutuhkan klien di masa mendatang. Selama tahap *implementation*, pengembang mengimplementasi *code* sesuai hasil desain lalu mengujinya.

Tahap *implementation* pada PXP berbeda dengan yang terdapat pada metodologi pengembangan. Pada PXP, proses pengujian dilakukan sebelum *code* program selesai. *System testing* adalah menguji semua fitur yang telah diimplementasikan pada tahap *implementation.* Selama *system testing* semua hasil implementasi diuji apakah sudah sesuai dengan kebutuhan klien. *Retrospective* menandakan tahap akhir iterasi dimana pengembang melakukan verifikasi apakah waktu implementasi sesuai dengan waktu estimasi. Klien dapat mengajukan perubahan pada iterasi yang sedang berlangsung maupun pada iterasi berikutnya. Proyek dianggap selesai jika hasil implementasi berhasil dijalankan tanpa *error* dan lulus *unit testing* atau kembali ke *iteration initialization* jika masih ada kebutuhan belum selesai maupun perubahan kebutuhan.

PXP merupakan turunan dari XP, sehingga praktik-praktik yang dilakukan pada XP secara tim disesuaikan agar dapat digunakan oleh pengembang tunggal. XP memiliki 12 praktik dasar yang digunakan dalam proyek pengembangan *software*. *Mapping* terhadap praktik XP dan PXP dilakukan untuk membandingkan antara praktik ketika dalam tim dan pengembang tunggal [6].

Tabel 3.1. *Mapping* praktik XP dan PXP [6].

Dua Belas Praktik	***Extreme Prograsmming***	***Personal Extreme Programming***
The Planning Game	Aktivitas dalam *planning game* adalah *writing-estimation-prioritization* dilakukan oleh tim pengembang dan klien untuk menentukan fitur-fitur utama yang menunjang proses bisnis klien.	Pengembang tunggal dapat memiliki banyak peran mulai dari menulis *user story,* estimasi, dan penentuan prioritas. *User story* berguna untuk memecah proyek besar menjadi bagian lebih kecil sehingga mudah bagi pengembang tunggal.
Small Releases	Fitur utama bagi klien akan diimplementasikan lebih awal dan diperbarui dalam jangka waktu pendek.	Rilis jangka pendek juga dapat dilakukan oleh seorang pengembang tunggal sama seperti pengembang dalam tim.
Metaphor	Setiap proyek memiliki penamaan khusus untuk menggambarkan proses kerja sistem guna membantu tim pengembang dan klien dalam memahami sistem yang dibangun.	Seorang pengembang tunggal dapat menentukan penggambaran sistem yang paling mudah dipahami oleh klien dan pengembang itu sendiri.
Simple Design	Membuat desain sistem sederhana mudah dipahami oleh seluruh member tim selama memenuhi kriteria kebutuhan klien tanpa menambahkan desain tambahan.	Pengembang tunggal mudah untuk membuat desain yang lebih sederhana dan lebih baik namun tetap mencakup semua fungsi sistem.

Testing	*Test-driven development* adalah kelebihan utama dari XP dimana dilakukan pengujian pada sistem secara terus-menerus. Sebelum menambah fitur baru, dilakukan pengujian untuk memverifikasi kesesuaian sistem yang dibangun.	Sebelum membuat modul sistem baru, pengembang harus membuat *interface* dahulu, lalu membuat *unit test*. Pengembang tunggal harus menguji dan meng*compile* tiap *code* modul sebelum melakukan proses *refactor*.
Refactoring	*Refactor* adalah teknik untuk memperbaiki struktur *code* program yang dilakukan dalam tahapan pendek.	Seorang pengembang tunggal dapat melakukan *refactoring* pada seluruh *code* yang telah ditulis, kecuali pada *code* yang bukan milik pengembang tersebut.
Pair Programming	Dua orang memprogram satu *code* modul bersamaan memungkinkan *code* dikoreksi sambil ditulis, hal ini menghasilkan *code* berkualitas tinggi tanpa mengurangi produktifitas.	Keuntungan *pair programming* otomatis tidak didapatkan oleh pengembang yang bekerja sendiri, namun dapat digantikan dengan meminta bantuan dari rekan serta dapat menerapkan prinsip *test first* selama membangun sistem.
Collective Code Ownership	Satu *code* untuk semua member tim, dengan prinsip ini menjadikan member dapat mengubah *code* proyek kapanpun. Hal ini dapat memicu ide-ide baru	Semua *code* adalah milik pribadi, namun seorang pengembang tunggal tidak akan mendapat keuntungan bertukar *code* seperti ketika dalam tim. Pengembang tunggal dapat

	dari member tim.	menyimpan *code* dalam *repository*.
Continuous Integration	*Continuous integration* bertujuan untuk mencegah atau mengurangi *code* guna tidak menyimpang dari basis *code*. Semua *code* member tim diintegrasi sekali sehari dengan basis *code* dan dilakukan pengujian menggunakan *test cases* oleh tim.	Semua *code* ditulis sendiri oleh pengembang tunggal sehingga tidak ada perubahan dari pihak luar. Potensi penyimpangan *code* tetap akan terjadi, sehingga *continuous integration* tetap harus dilakukan secara berkala.
40-Hour Week	Pemrogram dalam proyek XP memiliki waktu kerja 40 jam tiap minggu guna mempertahankan produktifitas dan kualitas kerja.	Pengembang tunggal dapat menganut waktu kerja 40 jam tiap minggu, pengembang dapat menghentikan pekerjaan jika produktifitas kerja berkurang akibat stres dan lelah.
On-Site Customer	Klien diminta untuk ambil peran bersama tim pengembang. Hal ini guna membantu tim pengembang melalui komunikasi intensif untuk membangun sistem yang sesuai kebutuhan klien.	Pengembang tunggal dapat melakukan komunikasi dengan klien via email atau telepon agar hubungan komunikasi dan *feedback* dari klien tetap berjalan.
Coding Standards	Semua member tim menggunakan standar *code* yang sama sehingga mempermudah ketika kerja	Pengembang tunggal dapat memilih gaya *coding* secara pribadi sebagai *coding standards* agar konsistensi tetap terjaga.

	bersama dan berbagi *code* antar member tim.	

Tabel 3.1 adalah *mapping* perbandingan antara praktik pada XP dan PXP. Berbeda dengan metodologi konvensional dimana siklus membangun *software* berjalan secara *sequential* berupa alur yang sudah ditentukan dan harus dilalui dimana tahap sebelumnya harus selesai untuk dapat berlanjut melakukan tahap berikutnya. Pada PXP, proses membangun aplikasi dimulai dengan melakukan pengumpulan data kebutuhan atau *requirements* pada Kejari Batu dan dilanjutkan dengan perencanaan atau *planning* menggunakan data yang telah diperoleh.

3.1 *Requirements*

Pada tahap ini dilakukan proses pengumpulan data kebutuhan di perpustakaan Kejari Batu. Proses pengumpulan data dilakukan dengan cara diskusi bersama pengurus perpustakaan Kejari Batu tanggal 12 Februari 2018. Selama proses diskusi, klien diminta untuk menyebutkan dan menjabarkan proses bisnis serta permasalahan yang terdapat pada perpustakaan Kejari Batu. Permasalahan yang ditemukan dari segi penyediaan buku, proses meminjam buku, proses mengembalikan buku, denda keterlambatan, keanggotaan, dan rekapitulasi daftar buku dan transaksi peminjaman. Berdasarkan permasalahan tersebut, klien kemudian diarahkan untuk mendiskripsikan kebutuhan-kebutuhan yang nantinya dapat diimplementasikan menjadi fitur-fitur dalam membangun aplikasi perpustakaan guna membantu menyelesaikan permasalahan yang ada pada perpustakaan Kejari Batu.

Semua kebutuhan yang telah diperoleh, masing-masing ditulis diatas kartu sesuai format "*Sebagai [peran] Saya dapat [tindakan] sehingga [tujuan]"* [15]. Penulisan *user story* yang baik dan benar sesuai bab 2 subbab 2.3 adalah bersifat *negotiable, estimatable,* dan testable. Salah satu contoh hasil penulisan kebutuhan ditunjukkan pada gambar 3.2. Didapatkan 16 *user story card* dari proses diskusi seperti ditunjukkan pada lampiran 1.

Nama *User Story*	Mendaftar anggota	*Priority*
Posisi	pengguna	*Size*
Saya ingin	Pengguna dapat daftar sebagai anggota	
Sehingga	Memiliki hak akses untuk meminjam buku	

Gambar 3.1. *User Story Card.*

Pada gambar 3.2 *user story card* terdiri atas nama *user story*, kemudian posisi klien, tindakan yang ingin dilakukan klien, dan tujuan yang ingin dicapai setelah melakukan tindakan tersebut. *Priority* adalah prioritas dari *user story* tersebut, sedangkan *size* adalah waktu yang dibutuhkan untuk implementasi *user story* dalam hitungan hari. Kebutuhan klien yang ditulis dalam *user story card* merupakan fitur-fitur dan fungsionalitas sistem pada aplikasi perpustakaan Kejari Batu yang akan dibangun. Dilakukan perencanaan lebih lanjut pada *user story* sebelum dapat diimplementasikan.

3.2 *Planning*

Berdasarkan pada hasil yang diperoleh dari tahap *requirements*, pada tahap ini dilakukan *planning* pada masing-masing *user story* dengan cara diskusi bersama klien. Tahap *planning* bertujuan untuk menentukan urutan prioritas *user story*, waktu yang dibutuhkan untuk implementasi *user story*, dan jumlah iterasi yang dibutuhkan hingga pengembangan aplikasi perpustakaan Kejari Batu selesai. Diskusi terdiri atas beberapa langkah yaitu estimasi *user story*, menentukan prioritas *user story*, dan menentukan *release planning*.

3.2.1 Estimasi *User Story*

Tahap estimasi bertujuan untuk memerkirakan waktu implementasi masing-masing *user story*. Estimasi berupa *story point* dan waktu implementasi dilakukan oleh pengembang. *Story point* digunakan sebagai nilai yang menunjukkan tingkat kesulitan implementasi, semakin besar *story point* maka waktu implementasi menjadi lebih lama. Hasil estimasi masing-masing *user story* ditunjukkan pada tabel 3.2.

Tabel 3.2. Hasil Estimasi *User Story.*

User Story	***Story point***
Pesan pengingat otomatis	1
Meng*input* peminjaman buku	1
Mengetahui jatuh tempo	1
Mendaftarkan buku	1
Mengembalikan buku	1
Pencarian buku berdasarkan judul	1
Akses khusus pengguna	1
Hitung denda otomatis	1
Notifikasi jatuh tempo peminjam dari sisi admin	1
Notifikasi buku baru	1
Mendaftar anggota	1
Mengubah data informasi website	1
Mengusulkan buku baru	1
Mengunggah file buku	1
Melihat daftar buku baru	1
Mendaftar admin baru	1

Tabel 3.2 berisi daftar semua *user story* terdiri atas judul *user story* dan *story point* pada tiap *user story*. *Story point* bernilai 1 untuk tiap *user story* yang diperoleh dari tahap *requirements*. Estimasi waktu implementasi tiap *user story* dengan nilai *story point* 1 dapat diselesaikan dalam waktu 3 hari.

3.2.2 Menentukan Prioritas *User Story* Menggunakan MoSCoW

Pada tahap ini pengembang dan klien melakukan diskusi kembali untuk menentukan prioritas tiap *user story*. Penentuan prioritas menggunakan metode MoSCoW. Tujuan metode ini untuk membuat proyek fokus pada kebutuhan utama klien dan mengesampingkan kebutuhan yang dianggap kurang berdampak besar pada proses bisnis klien maupun aplikasi perpustakaan Kejari Batu. Hal tersebut guna membatasi kebutuhan klien, sehingga mengurangi waktu penyelesaian proyek. *User story* akan dikelompokkan kedalam 4 kategori yaitu, *Must have*, *Should have*, *Could have*, dan *Won't have* [20].

Prioritas *user story* ditentukan berdasarkan kriteria *risk* dan *business value* [21][22]. *Risk* diartikan sebagai suatu *user story* apabila tidak diimplementasikan maka proyek pengembangan dianggap gagal. *Business value* adalah *user story* memiliki nilai yang dapat menunjang kinerja instansi atau klien dalam proses bisnis.

1. *Must have*, adalah *user story* yang memenuhi kriteria *business value* dan *risk*.
    a. Mendaftar anggota, *user story* ini penting dan harus diimplementasikan pada perpustakaan, karena proses bisnis meminjam buku, seorang pengguna harus terdaftar sebagai anggota.
    b. Mendaftar buku, *user story* ini menjadi fungsi utama dan harus ada karena membantu proses bisnis klien yaitu menyimpan dan *backup* data buku, pencarian, dan meminjam buku.
    c. Peminjaman buku, *user story* ini menjadi fitur utama aplikasi perpustakaan untuk proses *input* data, menyimpan dan *backup* secara terkomputerisasi. Fitur ini dibutuhkan untuk notifikasi jatuh tempo.
    d. Pesan pengingat otomatis, *user story* ini menjadi fitur utama yang harus ada pada aplikasi untuk mencegah anggota terlambat mengembalikan buku.
    e. Pencarian buku berdasarkan judul, *user story* ini menjadi fitur utama karena membantu pengguna mencari ketersediaan buku yang akan dipinjam.
    f. Hitung denda otomatis, *user story* ini menjadi fitur utama yang harus ada untuk membantu admin dalam menghitung biaya denda keterlambatan pengembalian buku.

g. Notifikasi jatuh tempo pinjam pada sisi admin, *user story* ini menjadi fitur utama yang harus ada untuk membantu admin melihat daftar peminjaman yang telah masuk jatuh tempo yang mana sulit dilakukan secara manual.
h. Notifikasi buku baru, *user story* ini menjadi fungsi utama dan harus ada pada aplikasi untuk membantu pengguna mengetahui daftar buku terbaru.
i. Akses khusus pengguna, *user story* ini menjadi fitur utama dan harus ada guna membatasi dan pembeda hak akses pengguna.
j. Mengembalikan buku, *user story* ini harus ada sebagai fitur utama proses bisnis meminjam buku pada aplikasi perpustakaan.
k. Mendaftar admin baru, *user story* ini harus ada sebagai fitur utama dan harus ada guna menyimpan data admin dan hak akses untuk masuk halaman kelola data aplikasi perpustakaan.

2. *Should have,* adalah *user story* yang tidak memenuhi kriteria *risk*, namun masih memiliki *business value* atau dapat membantu proses bisnis pengguna perpustakaan. *User story* pada kategori *should have* diimplementasikan dalam kurun waktu yang ditentukan, akan tetapi proyek tidak dianggap gagal meski tidak diimplementasikan. *User story* yang masuk pada kategori ini adalah sebagai berikut:
   a. Mengubah data informasi website, *user story* ini membantu pengguna menyediakan informasi perpustakaan dan Kejari Batu, seperti logo, nama instansi, alamat, dan kontak.
   b. Mengetahui jatuh tempo, *user story* ini dapat membantu pengguna melihat tanggal jatuh tempo peminjaman melalui aplikasi sehingga tidak lupa.
   c. Mengusulkan buku baru, *user story* ini sebagai fitur untuk membantu anggota mengirim usulan buku melalui aplikasi perpustakaan untuk melengkapi koleksi buku perpustakaan.
   d. Mengunggah file buku, *user story* in i sebagai fitur penting untuk membantu pengguna menyimpan berkas buku dalam bentuk *softcopy* kedalam sistem.
3. *Could have,* adalah *user story* yang tidak masuk dalam kriteria *risk* atau *business value*, namun dapat membantu melengkapi fitur aplikasi yaitu *user story* melihat daftar buku baru.

4. *Wouldn't have,* dari hasil diskusi kebutuhan yang telah dilakukan antara pengembang dan klien, tidak didapatkan *user story* yang masuk dalam kategori *wouldn't have.*

Hasil semua *user story*, estimasi, dan penentuan prioritas pada masing-masing *user story* ditunjukkan pada tabel 3.3.

Tabel 3.3. Daftar User Story Hasil Penentuan Prioritas Moscow.

User Story	***Story point***	***Priority***
Mendaftar anggota	1	*Must have*
Peminjaman buku	1	*Must have*
Mendaftarkan buku	1	*Must have*
Pesan pengingat otomatis	1	*Must have*
Pencarian buku berdasarkan judul	1	*Must have*
Hitung denda otomatis	1	*Must have*
Notifikasi jatuh tempo peminjam dari sisi admin	1	*Must have*
Notifikasi buku baru	1	*Must have*
Akses khusus pengguna	1	*Must have*
Mengembalikan buku	1	*Must have*
Mendaftar admin baru	1	*Must have*
Mengubah data informasi website	1	*Should have*
Mengetahui jatuh tempo	1	*Should have*
Mengusulkan buku baru	1	*Should have*
Mengunggah file buku	1	*Should have*
Melihat daftar buku baru	1	*Could have*

Tabel 3.4 menunjukkan daftar *user story* hasil estimasi dan penentuan prioritas. *Priority* adalah prioritas pada tiap *user story* yang ditunjukkan dalam urutan *must have*, *should have*, dan *could have*. Tahap berikutnya adalah membuat *release planning* menggunakan daftar *user story* pada tabel 3.2. Seorang klien selain diperbolehkan menambahkan *user story* sewaktu-waktu, klien juga diperbolehkan mengubah prioritas *user story* tersebut.

3.3 *Release Planning*

Berdasarkan pada hasil estimasi dan penentuan prioritas *user story,* maka tahap *planning* dilanjutkan dengan membuat daftar *release planning*. *Release planning* berisi tugas-tugas yang akan dikerjakan pengembang dalam membangun aplikasi perpustakaan Kejari Batu. Pada tahap ini pengembang bersama klien menentukan *user story* mana yang akan diimplementasikan terlebih dahulu. Proses pada PXP berupa perulangan atau iterasi-iterasi. Satu iterasi ditentukan dengan cara mengalokasikan *user story* sesuai urutan prioritas. *Story point* tiap *user story* dijumlahkan hingga sama dengan nilai *velocity*. Nilai *velocity* dapat diperoleh dari dokumen proyek serupa yang pernah dilakukan, jika tidak ada maka nilai *velocity* bisa ditentukan dengan cara memperkirakan nilai *velocity* itu sendiri. Nilai *velocity* ditentukan sendiri oleh pihak pengembang.

User story yang telah dijumlahkan tersebut membentuk iterasi ke-1. Jika terdapat sisa *user story*, maka *user story* tersebut akan dijumlahkan lagi hingga bernilai sama dengan *velocity,* lalu memasukkan *user story* tersebut kedalam iterasi baru. Proses ini dilakukan secara berulang hingga tidak ada sisa *user story*. Hasil *release planning* ditunjukkan pada tabel 3.4.

Tabel 3.4. Daftar Release Planning.

Iterasi 1		
User Story	***Priority***	***Story point***
Mendaftar anggota	*Must have*	1
Mendaftarkan buku	*Must have*	1
Meng*input* peminjaman buku	*Must have*	1
Pesan pengingat otomatis	*Must have*	1
Velocity		**4**
Iterasi 2		
User Story	***Priority***	***Story point***
Pencarian buku berdasarkan judul	*Must have*	1
Hitung denda otomatis	*Must have*	1
Menerima notifikasi buku baru	*Must have*	1
Melihat daftar jatuh tempo peminjam dari sisi admin	*Must have*	1

Velocity		**4**
Iterasi 3		
User Story	***Priority***	***Story point***
Mendaftar admin baru	*Must have*	1
Akses khusus pengguna	*Must have*	1
Mengembalikan buku	*Must have*	1
Mengubah data informasi website	*Should have*	1
Velocity		**4**
Iterasi 4		
User Story	***Priority***	***Story point***
Mengusulkan buku baru	*Should have*	1
Mengunggah file buku	*Should have*	1
Melihat jatuh tempo	*Should have*	1
Melihat daftar buku baru	*Could have*	1
Velocity		**4**

Pada tabel 3.4 nilai *velocity* ditentukan bernilai 4, sehingga dalam satu kali iterasi pengembang dapat mengerjakan 4 *story point* yang diselesaikan dalam waktu 12 hari. Satu iterasi berisi judul tiap *user story* sesuai urutan prioritas *user story* yang telah ditentukan. Berdasarkan tabel 3.4, diketahui bahwa *user story point* berjumlah 16, sehingga diperlukan 4 kali iterasi untuk dapat menyelesaikan seluruh *user story.* Proyek pengembangan aplikasi perpustakaan Kejari Batu direncanakan selesai selama 48 hari.

Dimungkinkan seorang klien untuk mengajukan tambahan kebutuhan dengan menulis *user story* baru. *User story* dapat ditambahkan ditengah-tengah proyek pengembangan aplikas. *User story* tersebut dilakukan estimasi oleh pengembang. Klien dapat memasukkan *user story* tersebut kedalam iterasi yang sedang berlangsung atau iterasi berikutnya.

Berdasarkan pada hasil daftar *release planning,* maka *user story* dapat mulai diimplementasikan dalam tahap *iteration development*. Tahap *iteration development* dilakukan secara berulang dalam proses implementasi *user story*

menjadi fitur-fitur pada aplikasi perpustakaan Kejari Batu. Dilakukan implementasi *user story* sesuai daftar *release planning*. Langkah-langkah yang dilakukan dalam *iteration development* yaitu *iteration initialization, design, implementation, system testing,* dan *retrospective*.

a. *Iteration Initialization*

Tahap ini menandakan awal *iteration development*. Pada tahap ini dilakukan pemilihan *user story* yang akan di implementasikan selama iterasi. *User story* dipilih berdasarkan prioritas dengan urutan *must have, should have, could have,* dan *wouldn't have* sesuai daftar iterasi dalam hasil *release planning*.

b. *Design*

Selama tahap *design*, pengembang membuat desain untuk semua *user story* dalam membangun aplikasi perpustakaan Kejari Batu yang telah dipilih pada *iteration initialization*. Mengacu pada penjelasan awal tahapan PXP pada bab 3, desain dibuat secara sederhana untuk *user story* sesuai iterasi yang sedang berlangsung. Desain yang dibuat adalah skema tabel desain *database* untuk tiap *user story*. Desain *database* oleh pengembang diartikan sebagai proses bisnis, dibuat dalam skema berisi tabel, atribut, dan *constraint* antar tabel [23]. Pengembang membuat satu desain pada tiap iterasi.

c. *Implementation*

Mengacu pada perbedaan yang dijelaskan di tahapan PXP pada awal bab 3, selama tahap *implementation* dilakukan proses menulis *code* program berdasarkan skema tabel desain *database user story* yang diperoleh dari tahap *design*. Proses implementasi *user story* menggunakan pendekatan *Test Driven Development* (TDD). TDD diterapkan secara berulang pada masing-masing *user story* dalam iterasi yang sedang berlangsung. TDD terdiri atas tiga tahap dimulai dengan *unit testing, code generation,* dan *refactoring*.

Unit testing adalah pengujian fungsionalitas *code* program, pengujian ini berbeda dengan proses pengujian fungsionalitas pada metodologi pengembangan *software* lain dimana pengujian dilakukan setelah sistem selesai. Pada *unit testing* pengembang menulis sebagian *code* program di awal tahap implementasi lalu melakukan pengujian. *Unit testing* dibuat dan diuji secara otomatis menggunakan

library PHP unit. PHP unit adalah *library* yang digunakan untuk menguji *code* program dengan bahasa pemrograman PHP. *Code generation* dilakukan setelah *code* program tiap fitur lulus *unit testing* dimana selama tahap *code generation* pengembang melengkapi *code* program tiap fitur hingga selesai. Jika diperlukan maka akan dilakukan proses *refactoring* atau optimasi *code* program. Semua fitur hasil implementasi harus dapat di*compile* tanpa *error* untuk dilakukan pengujian pada tahap *system testing*.

d. *System testing*

Mengacu pada penjelasan awal tahap PXP pada bab 3, dalam tahap ini dilakukan pengujian pada hasil implementasi. Proses pengujian dilakukan oleh pihak pengurus perpustakaan Kejari Batu dengan didampingi pengembang. Klien menentukan kriteria untuk tiap fitur hasil implementasi *user story* dengan tujuan mendapatkan kecocokan antara sistem yang dihasilkan dengan kebutuhan awal pada saat tahap *requirements* dan *planning*. Pengujian menggunakan *user acceptance test*. *User acceptance test* dapat diartikan sebagai pengujian *black box.* Pada PXP, seorang klien membuat *user acceptance test* untuk tiap *user story* dalam iterasi yang sedang berlangsung. Satu *user story* dapat memiliki lebih dari satu *user acceptance test*.

e. *Retrospective*

Pada tahap *retrospective* dilakukan verifikasi pada semua *user story* yang telah diimplementasikan dan diuji dalam tahap *system testing*. Verifikasi berupa perbandingan waktu estimasi dengan waktu realisasi. Jika ditemukan perbedaan waktu realisasi dengan waktu estimasi, maka dicari penyebab terjadi *under* atau *over* estimasi tersebut untuk mencegah potensi untuk terjadi perbedaan waktu pada proyek mendatang.